# CR1M3: The CyberCrime Handbook

This book covers a wide variety of techniques, tactics, and technical issues related to CyberCrime and security. It may be of interest to both offensive and defensive players in CyberCrime or security.

It is a mix of technical details about attacks, tutorials on related software, details about markets and how they operate, and stories related to CyberCrime.

## Early Reader Program

This book is currently under development. By purchasing a copy you'll be guaranteed all updates as they are released on LeanPub. I'm aiming to release at a minimum 2 updates every month. As part of the early reader program you'll get:

- Cheaper price than buying when it's finished
- At least 2 updates a month the book
- Private email newsletter
- Access to my private Telegram server
- Ability to provide feedback and make requests about content
- All the content at https://CR1M3.com and more

If you've gained access to this PDF outside the early reader program and would like to join you can do so by going to https://leanpub.com/CR1M3.

## Table of Contents

# 1. Cyber Intrusion - Common Attacks

## 1.1 Man in the Middle (MiTM)

A man in the middle attack (MiTM) is one in which the attacker positions himself in between a legitimate website and a client side user. It is one of the oldest forms of attack used on the internet.

### ARP Spoofing

One of the early ways MiTM was performed was called ARP spoofing or ARP poisoning. This involved connecting a computer to a network such as a public wifi network, and then announcing to the network that it is the router. This would result in other devices on the network sending their outgoing internet requests to the attackers computer. The attacker then sends the packets on to their intended destination so that everything appears to be working normally for the victims on the network. By having traffic routed through the attackers computer they can then analyze this traffic and extract sensitive information.

As a website owner one of the best ways to protect against a ARP Spoofing attack is to simply make use of an SSL certificate and enforce HTTPS. This can be done using a program like LetsEncrypt to create a free SSL certificate or buying one from a certificate authority (CA). When HTTPS is used all packets are encrypted with the final destinations public key. This means that even if an attacker is able to intercept the traffic they won’t be able to read it.

## IP Spoofing

Attacker creates IP packets which have a false source IP address in order to impersonate another server. Can be used as part of a MiTM attack by sending false packets to both parties and forwarding on the responses. This gives the impression to both parties that they’re talking to each other when in actuality they are talking to the attacker.

## DNS Spoofing (DNS Cache Poisoining)

An attack that involves corrupting a Domain Name Server so that website names can be associated with an incorrect IP address, thus allowing for MiTM attacks or phishing.

## SSL Stripping

Involves intercepting the user when they are being transferred from an open HTTP connection to a secure HTTPS connection. The attacker makes a secure connection with the targets intended destination for example “https://website.com”. From the end websites point of view the website is being delivered via HTTPS. The attacker then forwards the result as an HTTP unsecure version of “http://website.com”.

SSL Stripping was first exposed as a method of attack in 2009, by security research Moxie Marlinspike at the annual BlackHat conference in Las Vegas.

## SSL Hijacking

SSL Hijacking is very similar to SSL Stripping, the main difference being that instead of forwarding an unencrypted version of the website to the target over HTTP, a fake certificate is used so the victim sees HTTPS.

The attacker provides a false SSL certificate to the end client making them interpret this SSL certificate as being that of the intended end website when it is in fact a certificate controlled by the attacker. The end-user encrypts their traffic using the fake SSL certificate, the data is decrypted by the attacker and then re-encrypted with the real private-key. The end website then responds, attacker receives the response, decrypts it, encrypts the data with the fake certificate and then forwards the result on to the victims computer.

One of the ways SSL Hijacking is averted is through the use of Certificate Authorities (CA). Certificate Authorities sign the public keys which are used by websites. These trusted root certificate authorities are shipped with software like mobile phones and web browsers. Fake certificates created by attackers are not signed by the appropriate trusted root certificate authorities and thus are detected by browsers.

### Superfish and SSL Hijacking

One previously popular method of SSL Hijacking involved something called Superfish. Superfish involved a certified root certificate authority *(now bankrupt, due to the incident)*, which published a fraudulent certificate for Google. The result was that the owner of the fake certificate could perform a SSL Hijacking attack on anyone who visited Google and had the root certificate CA installed on their device. Since the problem was exposed their has been a major effort to remove the root CA certificate from software and devices. You can check if your device is vulnerable to Superfish or other bad trusted authorities by visiting badSSL.com.

Originally the attack could only be performed by the holder of the private key. However the key was later leaked meaning anyone could perform the attack on a device which had the bad root CA installed.

Well Superfish has been removed on most devices it demonstrates the weakness of SSL. Your HTTPS connection is only as trusted as the root certificates on your device. When it comes to government organizations for example, there is not much guarantee that they wouldn’t be able to obtain fake keys *(or already have them)*.

## 1.5 Credential Stuffing

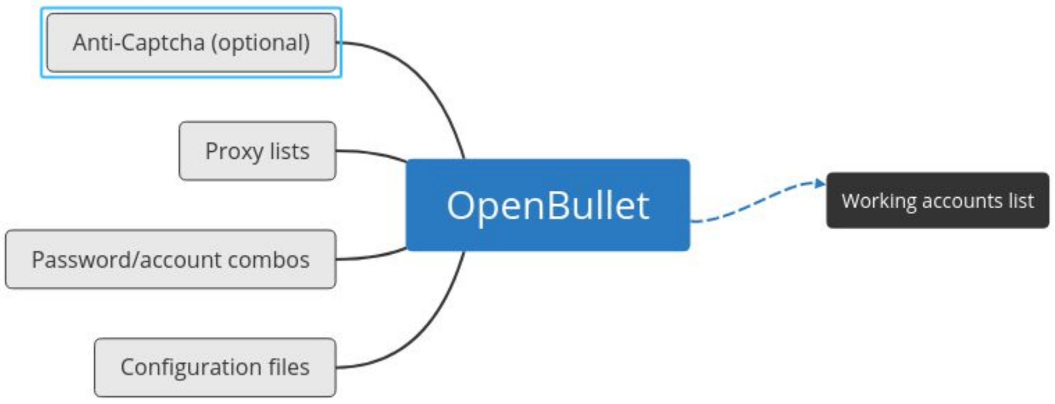
Credential stuffing is the use of leaked password sites to gain a list of valid username/password combinations for a target site.

Often leaked passwords come from large website breaches like the famous LinkedIn breach, but they can also come from smaller, less known sites. For example a small forum site with 100,000 users when attacked *(for instance by an SQL injection attack)* may leak a list of unprotected passwords.

While the small forum might not have much of value for the attacker the account/password list provides the raw resources which can be used for a credential stuffing attack on several other sites.

To get a more in depth description of the process of doing a credential stuffing attack, see my paper on the topic.

Another attack that can be used in conjunction with credential stuffing is SSL certificate unpinning. This is where a certificate which has been compiled into a mobile app is extracted. Once the certificate has been extracted it can be used to make requests as a mobile device instead of through the normal web API. Often companies that have captcha on their website will not have captcha on the mobile app, thus by pretending to be a mobile app an attack can become much more efficient.



## SSL Pinning and Credential Stuffing

Another technique that can be used in conjunction with credential stuffing is SSL certificate unpinning. This is where a certificate which has been compiled into an app to prevent MiTM and packet sniffing is disabled. Once the certificate has been disabled the network traffic can be analyzed and that data used to create a configuration file. Often a configuration file based on a mobile app traffic is more efficient than one using website traffic. This is because often companies that require completing a captcha on their website don’t have the same requirment on their mobile app.

## Bad Captchas

Some servers only require changing a user-agent value to disable captcha completely where others are more complicated and secure.

One cracker I talked to reported that he had encountered fake captcha systems, which appeared to be Captcha but would actually accept any submitted value. On the front-end the captcha widget would simply generate a random number using the code snippet below:

```
<input value="0.4464311458655813" name="captcharand" id="captcharand"
style="display:inline-block" type="hidden">

<div style="display:inline-block;">

<span id="captcha">
  <img onclick="reloadimg(); return false;" src="captcha/generate.php"
style="cursor:pointer;">
</span>
<script language="javascript" type="text/javascript">
document.getElementById("login").focus();

function reloadimg() {
  var a = Math.random();
  document.getElementById("captcharand").value=a;
  document.getElementById("captcha").innerHTML = '<img
onclick="reloadimg();return false;"
src="captcha/generate.php?rand='+a+' "
style="cursor:pointer;" />';
};
reloadimg();
</script><br>Captcha:<br>
```

## Note: Password Hashing

A successful SQL attack can yield a list of hashed or unhashed passwords. Hashed passwords are ones which are transformed using a one-way function which transforms them into a standard sized string of scrambled characters. The same password input run through the same hashing algorithm will always produce the same hased result.

Website developers should ensure to always encrypt passwords as it could potentially protect users from having their passwords exposed. A leaked list of account/password combinations which are hashed can require a significant amount of effort to make use. A